**Основные структуры данных**

**Целочисленные типы**

Целочисленные типы: знаковые и беззнаковые.

Беззнаковые имеют диапазон от 0 до некоторой константы. Знаковые типы имеют почти симметричный диапазон отрицательных и положительных значений.

Все целочисленные значения характеризуются количеством байт, отводимых под хранение одного такого значения:

* short -215..215-1 (2 байта);
* unsigned short 0..216;
* int -231..231-1 (2-4 байта);
* unsigned int 0..232 (не менее 4 байта);
* long -231..231-1;
* unsigned long 0..232;
* long long -263..263-1 (8 байт);
* unsigned long long 0..264.

**Типы для чисел с плавающей точкой**

Кроме целых чисел имеются числа с плавающей точкой, которые хранят вещественные числа.

0,15 = 15\*10-2 =0,015\*10=…

Формула выглядит следующим образом:

, (1)

где m — мантисса (значение цифры числа, 0<m<=1);

Q — основание системы счисления;

p — порядок.

Знак (1 разряд); порядок (от 7 разрядов); мантисса (от 23 разрядов).

В C++ есть следующие типы для чисел с плавающей точкой:

* float (4 байта);
* double (8 байт);
* long double (не менее 8 байт).

Все эти типы различает точность представления числа и возможный диапазон значений.

Для них определены:

* сложение;
* вычитание;
* умножение;
* деление.

**Логический тип**

Логический тип bool (1 байт), в котором true — истина, false — ложь.

Операции следующие:

* ! (отрицание);
* && (конъюнкция (и));
* || (дизъюнкция (или)).

Основным источником логических значений являются операции сравнения:

* == (равно);
* != (не равно);
* > (больше);
* < (меньше);
* >= (больше или равно);
* <= (меньше или равно).

С++ автоматически может производить преобразования целого типа в логический и наоборот.

Значению false соответствует 0, а значению true — число отличное от нуля.

* bool f=128;//true;
* bool t=0; //false.

**Массивы**

Массив — это структура данных, которая позволяет хранить множество значений одного и того же типа.

Структура объявления массива принимает вид: <тип элемента> <имя массива> [<количество элементов>], к примеру, int a[100]; — массив а из 100 элементов типа int, где short b[12]; — массив b из 12 элементов типа short.

Размер массива может задаваться константой, но не переменной.

Обращение к отдельному элементу массива выглядит следующим образом: <имя массива> [<индексное выражение>], при этом номера элементов массива всегда начинаются с 0, к примеру, a – a[0]…a[99], b – b[0]…b[11].

В С++ есть возможность объявлять массив и инициализировать его. Это делается с помощью следующей структуры: <тип элемента> <имя массива> [<количество элементов>] = {<значения>}, к примеру, int b[5]={1,5,13,11}.

Можно инициализировать не весь массив, а только некоторые элементы, тогда оставшиеся элементы будут заполнены нулями: int a[100] ={1,2,3,4}, где a: 1 2 3 4 0..0.

Если необходимо весь массив инициализировать нулями, то можно указать пустые фигурные скобки: c[10]={} c: 0 0 0 0 0..0.

Для организации ввода используется следующий цикл:

const int n=10;

int a[n];

for (int i=0;i<n;i++)

cin>>a[i];

for (int i=0;i<n;i++)

cout<<a[i]<< « »;

Чтобы избавить пользователя от ручного ввода данных можно использовать генератор случайных чисел.

Генератор случайных чисел строит числа по определенному алгоритму, который зависит от некоторого стартового числа. Чтобы эти числа всегда были разными, необходимо каждый раз передавать в генератор случайных чисел уникальное значение. Таким уникальным значением является текущее время. Для работы с временем необходимо подключить библиотеку #include <ctime>.

С помощью команды srand(time(0)) осуществляется включение режима генерации случайных чисел (параметр — текущее время).

for (int i=0;i<n;i++)

a[i] = rand()%100 + 1; // в a[i] будет случайное число в диапазоне [1;100].

**Структура**

Структура — это специальный тип данных, который позволяет в одной переменной хранить данные различных типов. Структура позволяет моделировать в программе объекты реального мира и затем унифицировано (однотипно) их обрабатывать.

Например, с помощью структур можно в одной переменной сохранить такие сведения о студенте, как его ФИО, пол, возраст, номер группы и, используя всего одну переменную, работать со всеми этими данными.

Встроенного типа «Структура» в С++ нет, поэтому для того, чтобы работать со структурами программисту необходимо использовать специальную языковую конструкцию для объявления различных структур.

struct <имя структуры>

{ <тип> <поле 1>;

<тип> <поле 2>;

…………………..

<тип> <поле n>;

};

Пример.

struct human

{

char name[200];

int age;

double height;

};

Поля структуры могут иметь любой тип С++, в том числе быть массивами и структурами. Доступ к полям структуры осуществляется через уточненные имена, то есть имена вида: <имя структур> <имя поля>.

Отдельные поля структуры могут использоваться так же, как и обычные переменные этого типа.

После определения структуры мы можем использовать имя этой структуры для объявления переменных: human h1, h2; каждая переменная h1 и h2 состоит из трех полей, которые объявлены в структуре human.

Все действия над структурами производятся отдельно над ее полями. Например:

cin>>h1, age;

h1, height+=3;

h1={«Anna», 18, 170};

Объявление структуры может располагаться как внутри метода main, так и вне его. Однако, рекомендуется располагать объявление структуры вне метода main, поскольку это позволит использовать структуру в различных функциях.

Очень часто в программах работают не с одной структурой, а с массивом структур.

struct human

{

char name[200];

int age;

double height;

}

int main()

{

const int n=5;

human a[n];

cout<< «введи информацию о людях» <<endl;

for (int i=0;i<n;i++)

{

cout << «введи имя»<<endl;

for (int j=0;j<200;j++)

cin>>a[i].name[j];

cout<< «введи возраст»<<endl;

cin>>a[i].age;

cout<< «введи рост»<<endl;

cin>> a[i].height;

}

//Найти количество людей, у которых рост выше заданного и возраст не оканчивается на 8.

int k=0;

cout<< «введи заданный возраст»<<endl;

double r;

cin>>r;

for (int i=0;i<n;i++)

if (a[i].height>r && a[i].age%10!=8)

k++;

cout<<k<<endl;

}

**Объединения**

Объединения в программировании — это объект, позволяющий нескольким переменным различных типов занимать один участок памяти.

В каждый момент времени объединение содержит один из нескольких элементов различных типов.

Структура объявления объединения:

union<имя объединения>

{ <тип> <поле 1>;

<тип> <поле 2>;

……………………

<тип> <поле n>; };

union all\_in\_one

{

int int\_value;

double double\_value;

long long\_value;

}

all\_in\_one p;

В переменной p может храниться значение типа long, double и int, но в разные моменты времени.

Размер памяти, который отводиться под такие переменные, равен размеру самого большого поля объединения.

p.int\_value=300;

cout<< «int value=» <<p.int\_value<<endl;

p.double\_value=3.14;

cout<< «double value=»<<p.double\_value<<endl;

Все переменные, которые использовались до этого, имели простой тип. Память под такие переменные выделялась компилятором автоматически, и компилятор сам отслеживал местоположение таких значений.

Чтобы явно получить адрес памяти, выделенный под такую переменную, необходимо применить операцию разыменования переменной. Данная операция обозначается как &.

Пример.

a — переменная, &а — адрес переменной.

int a=6;

cout<< «значение а=»<<a<<endl;

cout<< адрес a= «<<&a<<endl;

Выводит значение a=6, адрес a = 0x0043fd38.

При работе с обычными переменными компилятор всегда сам контролирует правильность использования некоторой области памяти.